IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Timothy L. Harris		
Title:	LINKED-LIST IMPLEMENTATION OF A DATA STRUCTURE WITH CONCURRENT NON-BLOCKING INSERT AND REMOVE OPERATIONS		
Application No.:	09/710,218	Filed:	November 10, 2000
Examiner:	Zhen, Li B.	Group Art Unit:	2194
Atty. Docket No.:	004-4896	Confirmation No.:	4731

October 28, 2005

Mail Stop Appeal Briefs - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF (37 C.F.R. § 41.37)

This brief is in furtherance of the Notice of Appeal, filed on April 27, 2005. The fee required under 37 C.F.R. § 41.20(b)(2) is provided in the accompanying Transmittal.

REAL PARTY IN INTEREST

The real party in interest in this appeal is Sun Microsystems, Inc., the assignee of record, as evidenced by the assignment recorded at Reel/Frame 011305/0889.

RELATED APPEALS AND INTERFERENCES

Appellant has no knowledge of any related appeals or interferences.

STATUS OF CLAIMS

Claims 1–30 are pending.

Of those, claims 1–6, 8–13 and 16–30 stand rejected. Claims 7, 14 and 15 are objected to as dependent on a rejected base claim (*see* Advisory Action, mailed 28 March 2005, but were

indicated allowable if rewritten in independent form. An amendment under 37 C.F.R. § 41.33(b) accompanies this brief and rewrites claim 7 in independent form.

Rejections of claims 1–6, 8–30 are therefore the subject of this appeal and claims 1–30, including the claims indicated allowable, are reproduced in the Appendix attached hereto.

STATUS OF AMENDMENTS

An amendment under 37 C.F.R. § 41.33(b) accompanies this brief, but has not yet been entered. That amendment rewrites claim 7 in independent form.

SUMMARY OF CLAIMED SUBJECT MATTER

The presently claimed subject matter relates to a simple and therefore highly usable non-blocking implementation of a concurrent shared object representation. Such an implementation of a concurrent shared object representation employs a single-target synchronization primitive to encode a marked node indication signifying logical deletion of one or more values from the shared object representation. *See* p. 3, lines 8 – 17. Although example implementations refer to a compare-and-swap (CAS) primitive, other primitives such as a locked load/store-and-compare (LL/SC) instruction pair may be employed in some variations.

Claim 1 is directed to a non-blocking concurrent shared object representation comprising a linked-list of nodes encoding of a group of zero or more values. *See* Figures 1, 2, 3A, 3B, and 11A – 11C. The shared object representation also comprises linearizable operations defined to implement semantics of at least insert and remove operations on the group. *See* Figures 4 – 10; p. 9, lines 9 – 15; p. 14, line 17 – p.15, line 23; p. 25, line 7 – p. 29, line 2. Concurrent execution of the linearizable operations is mediated using a first synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group. *See* p. 11, line 25 – p. 12, line 5. The first synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination. *See* p. 9, lines 9 – 15.

Claim 16 is directed to a method of managing access to a linked-list of nodes susceptible to concurrent operations on a group encoded therein. *See* Figures 1, 2, 3A, 3B, and 11A – 11C.

The method comprises separating deletion of a value from the group into at least two functional sequences. *See* Figures 5 and 7 – 9; p. 13, line 18 – p. 14, line 16; p. 14, line 28 – p.16, line 27. The first functional sequence performs a logical deletion of the value using a synchronization primitive to mark a corresponding one of the nodes. *See* Figure 5; p. 14, line 28 – p. 15, line 23. The second functional sequence excises the marked node from the linked-list. *See* Figure 7; p. 13, line 18 – p. 14, line 16. The synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination. *See* p. 9, lines 9 – 15.

Claim 25 is directed to a computer program product encoded in at least one computer readable medium. The computer program product comprises at least two functional sequences providing non-blocking access to a concurrent shared object. *See* Figures 4 – 10; p. 9, lines 9 – 15; p. 14, line 17 – p.15, line 23; p. 25, line 7 – p. 29, line 2. The concurrent shared object is instantiable as a linked-list of nodes encoding of a group of zero or more values. *See* Figures 1, 2, 3A, 3B, and 11A – 11C. A first of the functional sequences is defined to implement semantics of an insert operation on the group. *See* Figure 4; p. 14, lines 17 – 27. A second of the functional sequences is defined to implement semantics of a remove operation on the group. *See* Figure 5; p. 14, line 28 – p. 15, line 23. Instances of the functional sequences are linearizable and concurrent execution thereof by plural processors of a multiprocessor is mediated using a synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group with separate physical excision of the corresponding node. *See* Figures 4 – 10; p. 9, lines 9 – 15; p. 14, line 17 – p.15, line 23; p. 25, line 7 – p. 29, line 2. The synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination. *See* p. 9, lines 9 – 15.

Claim 29 recites some elements in accordance with 35 U.S.C. § 112(6). In particular, claim 29 is directed to an apparatus comprising plural processors and one or more data stores addressable by each of the plural processors 102. *See* Figure 1; p. 9, line 16 – p. 10, line 23. The apparatus also comprises means for coordinating concurrent execution, by ones of the plural processors, of at least insert and remove operations on a group of zero or more values encoded in the one or more data stores. *See* Figures 1, 4, 5, and 7; p. 9, line 16 – p. 10, line 23; p. 14, lines 17 – 27; p. 14, line 28 – p. 15, line 23; p. 13, line 18 – p. 14, line 16 (corresponding structures include processors 102 programmed in accordance with the illustrated and described operations).

The coordinating employs a first synchronization primitive to encode an indication signifying logical deletion of a corresponding one of the values from the group and a second synchronization primitive to physically excise the node corresponding to the logically deleted value. *See* Figures 5, and 7; p. 14, line 28 – p. 15, line 23; p. 13, line 18 – p. 14, line 16. The synchronization primitives atomically examine and update their respective single targets, the updating being conditional on the examination. *See* p. 9, lines 9 – 15. The apparatus also comprises means for traversing the encoded group without use of an atomic operation. *See* Figure 1; Figure 6; p. 15, lines 24 – 28 (corresponding structures include processors 102 programmed in accordance with the find operations).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Ground I: Claims 1–6, 8, 10–13, 16–22 and 25–30 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "A Lock-Free Multiprocessor OS Kernel" by Massalin and Pu (*Massalin*) in view of U.S. Patent No. 6,651,146 to Srinivas et al. (*Srinivas*).

ARGUMENT

As a general proposition, obviousness is a legal determination based on underlying factual inquiries. *See* Minnesota Min. & Mfg. Co. v. Johnson & Johnson Orthopedics, Inc., 976 F.2d 1559, 24 U.S.P.Q.2d (BNA) 1321, 1332-1333 (Fed. Cir. 1992). Graham v. John Deere Co., 383 U.S. 1, 17 (1966) defines the factual inquiries utilized to evaluate the prior art. Specifically, the prior art is evaluated in terms of: (1) its scope and content; (2) the differences between the prior art and the claimed invention; (3) the level of ordinary skill in the art at the time the application was filed; and (4) objective, or secondary, evidence of nonobviousness such as commercial success, failure of others, long-felt need and unexpected results, which must be considered in reaching a conclusion of obviousness. Graham v. John Deere Co., 383 U.S. 1, 17, 148 U.S.P.Q. 459, 460 (1966); Panduit Corp. v. Dennison Mfg. Co., 810 F.2d 1561, 1566-67, 1 U.S.P.Q.2d 1593, 1595-96 (Fed. Cir. 1987); Minnesota Min. & Mfg. Co. v. Johnson & Johnson Orthopaedics, Inc., 976 F.2d 1559, 24 U.S.P.Q.2d 1321, 1333 (Fed. Cir. 1992).

In the present appeal, the issue relates to specific differences between the prior art and appealed claims. All claim limitations must be considered in the obviousness analysis. Indeed,

it is clear error to ignore limitations clearly set forth in the claims. *See* Panduit Corp., 1 U.S.P.Q.2d at 1603–04, 810 F.2d at 1576. Here the Office has ignored clear limitations of the claims.

In rejecting claims under 35 U.S.C. § 103, the examiner bears the initial burden of presenting a *prima facie* case of obviousness. *See* In re Rijckaert, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d (BNA) 1955, 1956 (Fed. Cir. 1993). A *prima facie* case of obviousness is established by presenting evidence that would have led one of ordinary skill in the art to combine the relevant teachings of the references to arrive at the claimed invention. *See* In re Fine, 837 F.2d 1071, 1074, 5 U.S.P.Q.2d (BNA) 1596, 1598 (Fed. Cir. 1988), In re Lintner, 458 F.2d 1013, 1016, 173 U.S.P.Q. (BNA) 560, 562 (CCPA 1972).

Appellant's Contentions with respect to Ground I

Claims 1–6, 8, 10–13, 16–22 and 25–30 were rejected under 35 U.S.C. §103(a) as being unpatentable over "A Lock-Free Multiprocessor OS Kernel" by Massalin and Pu (hereinafter *Massalin*) in view of U.S. Patent No. 6,651,146 granted to Srinivas et al. (hereinafter *Srinivas*). In making the rejections, the Office has assumed a scope of disclosure for *Massalin* that is simply inconsistent with the actual teachings of *Massalin*. In particular, the Office amasses various unrelated descriptive snippets from *Massalin* and reconstitutes a new non-blocking linked-list implementation (and operations thereon) never contemplated by *Massalin*, using Appellants' disclosure and claims as a guide. In doing so, the Office misrepresents the disclosure of *Massalin*, fails to consider Appellants' claims "as a whole", or both. In either case, it is legal error.

In making the assessment of differences between the prior art and the claimed subject matter, section 103 specifically requires consideration of the claimed invention "as a whole." Ruiz v. A.B. Chance Co., 357 F.3d 1270, 1275, 69 U.S.P.Q.2d (BNA) 1686, 1690 (Fed. Cir. 2004). Inventions typically are new combinations of existing principles or features. Envtl. Designs, Ltd. v. Union Oil Co., 713 F.2d 693, 698, 218 U.S.P.Q. (BNA) 865 (Fed. Cir. 1983) (noting that "virtually all [inventions] are combinations of old elements"). The "as a whole" instruction in title 35 prevents evaluation of the invention part by part. Ruiz, 357 F.3d at 1275, 69 U.S.P.Q.2d at 1690. Without this important requirement, an obviousness assessment might

successfully break an invention into its component parts, then find a prior art reference corresponding to each component or (as here) unrelated features of a single reference. This line of reasoning would import hindsight into the obviousness determination by using the invention as a roadmap to find its prior art components. Further, this improper method would discount the value of combining various existing features or principles in a new way to achieve a new result—often the essence of invention. Id.

Furthermore, even after reinventing the disclosure of *Massalin*, the Office finds that its creative "interpretation", which (unsupported by the actual disclosure of *Massalin*) posits use of a dual-target Compare-and-Swap in a deleted node marking role, fails to account for a *single-target* limitation as it relates to Appellant's claimed use of a synchronization primitive to effectuate logical deletion of a node. To address this defect, the Office relies on *Srinivas*, apparently for the proposition that CAS operations may be used in concurrent linked-list implementations. However, even assuming *arguendo* that the Office's interpretation of *Massalin* were proper, *Srinivas* fails to disclose or suggest use of a CAS (or other single-target synchronization operation) to mediate concurrent execution of insert and/or remove operations by encoding a marked node indication signifying logical deletion of a corresponding value.

For each of the appealed from rejections, the Office apparently bases its conclusion of obviousness on an assumption that replacing a *dual-target* Compare-and-Swap in the hypothecated (but not actually disclosed) role of coordinating a deleted node marking with a *single-target* synchronization is a mere matter of design choice. Such an assumption is, with all due respect, simply without factual basis. As a legal matter, absent proper basis for the assertion, there is no *prima facie* case of obviousness. Indeed, the idea that a dual-target Compare-and-Swap (DCAS) may simply be replaced with a single-target synchronization (*e.g.*, a CAS) flies in the face of nearly 20 years of research in the art.

In typical concurrent software implementations and consistent with *Massalin's* usage, a CAS operation is not simply interchangeable with a DCAS operation. Indeed, the very reason that a DCAS operation is employed in a given implementation is that a particular algorithmic approach to managing concurrency requires that two synchronization targets be managed atomically. One cannot merely replace a DCAS with a CAS operation (or for that matter with

two CAS operations) while maintaining any semblance of a pre-existing algorithmic approach to managing concurrency. Rather, if even possible, a CAS-based algorithm typically requires a different concurrency management techniques.

At the risk of over dramatizing the point, the Office's reasoning is a bit like relying an article describing Evil Kneviel's jump of the Snake River Canyon on a dual-wheeled motorcycle, observing that the reference does not teach use of a single-wheeled vehicle, but concluding that it would be obvious to employ a unicycle in the crossing.

Claims 1, 16, 25 and 29 – references fail to disclose or suggest use of single-target synchronization, as claimed

All appealed from rejections necessarily depend on the Office's "modification" of *Massalin* to instead employ CAS synchronization from *Srinivas*. Therefore, even leaving aside additional deficiencies (detailed below) in the Office's reliance on *Massalin* and *Srinivas*, the rejection of each independent claims (as well as those dependent therefrom) must be reversed if there is no teaching or suggestion to modify Massalin to employ CAS synchronization in place of a DCAS synchronization or if such a proposed modification would necessitate a change in *Massalin's* approach to managing concurrency.

Independent claim 1 recites mediating concurrent execution of the linearizable operations using a single target synchronization primitive to encode a marked node indication signifying logical deletion. See claim 1, (reciting "`wherein the first synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.`") This limitation is similarly recited in independent claims 16, 25, and 29. To address this limitation, the Examiner employs *Srinivas*. The Examiner contends that the claims can be achieved simply by replacing the `2CAS` employed by *Massalin* with a `CAS` from *Srinivas*. As motivation, the Examiner states that

> `It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of a synchronization primitive that atomically examines and updates a single target, the updating being conditional on the examination as taught by Srinivas to the invention of Massalin because primitive operation including an atomic operation of "compare and swap" (CAS) allow list management`

> without the use of locks [col. 5, lines 57 - 67 of Srinivas].

Final Office Action, ¶ 7, page 4 (detailing rationale in rejection of claim 16).

Massalin already includes disclosure of a CAS instruction, and the Office does not rely on *Srinivas* for anything more than disclosure of the CAS instruction. Since *Massalin* already disclosed a CAS instruction, the additional disclosure of a CAS instruction in *Srinivas* is duplicative (*Massalin* already disclosed CAS) and conflicts with any suggestion that it was desirable to simply substitute the instructions as performed by the Examiner.

> The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990)

Rather, a person of ordinary skill would immediately understand that the reason *Massalin* described use of a 2CAS, rather than the CAS instruction, which he found suitable for other purposes, was that the atomic management of two synchronization targets provided by the 2CAS instruction (and not by a CAS instruction) was necessary to his strategy for managing concurrency.

Moreover, simply substituting instructions trivializes the complexities of code necessary to implement any concurrency techniques consistent with *Massalin* and trivializes Applicant's claimed invention. Merely replacing the 2CAS of *Massalin* with a CAS does not achieve Applicant's claimed invention, and both the implementation and the result of such a change is unknown. The Examiner contends that replacing a 2CAS, which operates with 2 reference values, 2 new values, and 2 targets, can simply be replaced with a CAS, which operates with 1 reference value, 1 new value, and 1 target. There is no indication of how such a substitution can be implemented and how it will affect *Massalin's* technique.

> If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959).

The substitution performed by the Examiner in an attempt to achieve Applicant's claims cannot support the rejections because 1) there is no suggestion or motivation to modify *Massalin* in such a manner, and 2) the proposed substitution would change the principle of operation of *Massalin*, as well as requiring substantial modification to *Massalin* that cannot be done without the aid of Applicant's disclosure.

For at least these reasons, Applicant respectfully requests that the unsupportable rejections of the independent claims be reversed and that this Honorable Board direct allowance of pending claims.

Detailed Analysis of Massalin

Because relied upon aspects of *Massalin* are a bit muddled in the appealed from final rejection, a brief review of *Massalin*'s actual disclosure is warranted. In particular, Section 3.3 of *Massalin* discloses two implementations of operations on a linked list. The first, noted below as (1), is non-blocking, but operates only at end nodes of the list. The second, which operates on interior nodes and which is noted below as (3b), is **not non-blocking** in that it employs a lock, i.e., a "binary marker" in *Massalin*'s parlance, set on node entry (using a *two-word* Compare-and-Swap) to "sit on [the node]" thereby preventing another operation from manipulating a node that precedes a to-be-deleted node.

The two implementations are distinct and separate. Attributes of one do not convey to the other. In particular, the second implementation is a **blocking** implementation and there is simply no teaching or suggestion to use (in the second implementation) a single target synchronization for any purpose recited in Applicant's claims.

Turning to the specifics, *Massalin* discloses:

(1) A first implementation of a linked list that includes non-blocking operations for pushing (and popping) an element onto (and from) the list *only at an end (head or tail) thereof. See Massalin*, Figure 2 and accompanying description (first paragraph of Section 3.3).

While the operations are non-blocking, concurrent execution is not mediated using a first synchronization primitive to encode a marked node indication signifying logical deletion of anything. The "insert" operation uses a one-word CAS operation to push a new element onto the head of the list while guarding against a

conflicting concurrent update of the head pointer. The "delete" operation uses a two-word CAS2 operation to pop an element from the head of the list by atomically updating the head pointer and a first node's pointer to a second node so that the head pointer instead identifies the node that was "second" and the node that was "first" no longer points into the list. Use of the CAS2 operation guards against a conflicting concurrent update of either the head pointer or the "first" node's pointer to the "second" node.

- *No marked node indication signifying logical deletion is employed for any purpose.*

- *Despite somewhat confusion terminology, the operations of Figure 2 do not implement semantics of insert and remove since they cannot operate on interior nodes, rather they implement push and pop semantics.*

(2) Massalin notes that a more sophisticated implementation that would allow deletion of interior nodes is "much harder." Two approaches to the "much harder" problem are discussed; we note them below as (3a) and (3b).

(3a) A first approach is described as apparently advocated by another researcher, Herlihy, in which a two-word compare and swap is used to maintain node-specific reference counts in coordination with traversal operations. The reference does not suggest (and the Office does not propose) that Herlihy's technique anticipates or renders obvious any claim.

(3b) The second approach (i.e., the approach proposed by *Massalin* in the remaining paragraphs of Section 3.3) apparently forms the basis of the Office's rejection. In particular, *Massalin* proposes that, as an alternative to reference counting, it would be possible to implement a strategy in which an interior node is deleted "only when the permanence of the previous node is guaranteed." Deletion is performed in two steps: (1) marking a node for deletion and (2) "sit[ting]" on the previous node and deleting the marked node. To guarantee permanence of that previous node, *Massalin* employs a lock, i.e., a "binary marker[1]" in *Massalin*'s parlance, set on node entry (using a *two-word* Compare-and-Swap) to "sit on [the node]" thereby preventing another operation from manipulating a node that precedes a to-be-deleted node.

As a result,

- *While a marked node indication may be described, there is simply no disclosure of mediation of concurrent operations using a single-target synchronization primitive to encode a marked node indication. Indeed, the*

---

[1] *Massalin* is a bit confusing this point. A "binary marker" is employed as a lock to ensure permanence of a particular node. A different node is "marked for deletion." A dual target CAS is employed to set a lock, i.e., the "binary marker." No mechanism is described for marking a node for deletion and, indeed, it is not at all clear from *Massalin* where such a mark would be encoded or how any concurrency would be managed.

Office interprets Massalin as teaching use of a two-word Compare-and-Swap for node marking.

- *Massalin's description is of a blocking, not non-blocking, implementation.*

Based on this more complete description of *Massalin*, we now address additional deficiencies in the Office's rejections.

Claims 1 and 25 – References fail to disclose or suggest non-blocking techniques

Independent claims 1 and 25 each recite non-blocking techniques ("`[a] non-blocking concurrent shared object representation`," in claim 1; ("`at least two functional sequences providing non-blocking access to a concurrent shared object`," in claim 25). As established above, the relied upon implementation in *Massalin* employs a lock and is therefore not non-blocking.

Although the Examiner has disagreed (*see* Advisory Action) with identification of the binary marker technique as blocking, *Massalin* specifically states that:

> `We set the mark at the same time we enter the node using a two-word Compare-and-Swap. This is easier than incrementing a counter because we don't have to read the mark beforehand - it must be zero to allow entrance. Non-zero means that node is being visited so we skip to the next one repeating the test. P. 6, Section 3.3.`

If the marker is non-zero, then access to the marked node is blocked. It should be clear to persons of ordinary skill that *Massalin*'s technique is blocking since the binary marker (i.e., lock) must be zero to allow entrance. As a result, no competing thread can visit the node until the lock is released (i.e., until the binary marker is reset to zero).

Claim 1 recites a "non-blocking concurrent shared object" and claim 25 recites "at least two functional sequences providing non-blocking access to a concurrent shared object." *Massalin*'s technique is blocking. Neither *Srinivas* nor any other art relied upon by the Office disclose or suggest modifications to *Massalin*'s technique that would result in a non-blocking operation. Accordingly, no *prima facie* case of obviousness has been made out and claims 1 and 25 (as well as those dependent therefrom) are allowable for at least this reason as well.

In responding to Applicant's contention that Massalin fails to disclose or suggest linearizable operations and non-blocking access, the Examiner confuses both Massalin's disclosure and Applicant's contentions. In the Advisory Action, the Examiner blurs together Massalin's marking for deletion with the binary marker for access. This confusion leads the Examiner to further state that if "applicant's submission that Massalin's marking step is a locking technique is correct, then it appears the marking step of the claims is also a locking technique." The contention that Massalin discloses a blocking technique is made with specific reference to the binary marker technique disclosed by Massalin for accessing a node, which is distinct from the marking for deletion. The conclusion by the Examiner is incorrect and baseless.

Claim 29 – References fail to disclose or suggest means for traversing the encoded group without use of an atomic operation

Claim 29 positively recites "`means for traversing the encoded group without use of an atomic operation`." In rejecting claim 29, the Office simplistically assumes that mere traversal in *Massalin* could not involve synchronization; however, that assumption is simply not consistent with the disclosure of *Massalin*. As reviewed above, *Massalin* sets a lock to prevent other operations from visiting a node. For such a lock to have any operational effect, a traversing operation would necessarily check lock state using an atomic operation. Otherwise, the implementation would be non-operative for failure of the traversal to recognize a lock set by a concurrent access. Indeed, the `VisitNextNode()` operation illustrated in Figure 3 includes a `CAS2` operation to traverse while checking that a reference count remains unchanged. *Massalin*'s replacement of a reference count (multiple states) with a binary marker (2 states) does not obviate this check.

The Office's reasoning is detailed in the Advisory Action, where the Examiner states that:

> `[S]ince traversal and update is done separately, then the traversal is done without the use of an atomic operation. Generally, traversal of the link-list does not change the content of the list because traversal of the link-list is a method of looking for a particular node value, which means it is a read access. Read access doesn't require synchronize access because it is not making any changes to the list. The compare-and-swap operation (atomic`

> operation) is used to provide lock-free synchronized access when an updated to the link-list is required (i.e. insert and remove). Therefore, the traversal method of Massalin is performed without an atomic operation.

Regardless of separation of traversal and update, the traversal disclosed in *Massalin* utilizes an atomic operation. On page 6, in section 3.3, *Massalin* specifically states that a 2-word Compare-and-Swap is utilized to set a mark in the run-queues implementation upon entering a node, which occurs during traversal of a linked-list of nodes. *Massalin* then states that "[w]e omit the traversal code since the only difference from unsynchronized access is the Compare-and-Swap." *Massalin* p. 6, Section 3.3. It should be abundantly clear that Massalin employs an atomic operation when traversing a linked list to set a binary marker, which allegedly allows *Massalin* to traverse a linked-list in a multi-threaded environment without implementing Herlihy's reference counting technique for traversal. The Examiner's assertion is not supported by *Massalin*, and clearly contradicts the disclosure of *Massalin*.

Applicant requests that the faulty rejection be withdrawn and the claim 29 and those dependent therefrom be allowed.

Claim 4 – References fail to disclose or suggest reclamation of storage associated with an excised node being independent of linearizable operations

With regard to claim 4, the Office contends that *Massalin's* disclosure of separation of run-queue traversal from queue element update and column 6, lines 1 – 18 of *Srinivas* disclose claim 4. Claim 4 recites "wherein reclamation of storage associated with the excised node is independent of the linearizable operations." Separation of traversal and queue element update simply does not relate to independence of storage reclamation and the linearizable operations, which consistent with claim 1 above, include at least insert and remove operations. The section *Srinivas* relied upon by the Office is as follows:

> List managers are software routines that can vary depending on the types of lists managed and the defined management process. In one embodiment of the present invention a list is managed primarily with operations employed to add elements to and remove elements from a list. In particular, embodiments of the present invention employ the following functions; "add a data element to the front of a list" (ATLF), "add an element to the back of a list" (ATLB) and "remove a data element from the front of a list"

> (RFLF), each function implemented using an atomic operation of a CAS and as disclosed enable management of a list of the present invention in an SMP system without the use of locks. In one embodiment of the ATLF, an ATLB and an RFLF, on a singly linked list of free memory data elements, allows improved memory/cache management in an SMP system.

These sections of *Massalin* and *Srinivas* fail to even make reference to storage reclamation. More particularly, the independence of (i) linearizable operations on a list and (ii) reclamation of storage for a node excised from the list is simply not disclosed or suggested by *Massalin* or *Srinivas*.

The Office fails to identify any art that discloses or suggests the limitations of claim 4 and therefore fails to make out a *prima facie* case of obviousness. The Examiner maintains the rejection of claim 4 by simply quoting the claim and supplying the above references sections of *Massalin* and *Srinivas*. All limitations of the claim must be disclosed or suggested by the references. It is clear error to ignore limitations clearly set forth in the claims. Panduit Corp., 1 U.S.P.Q.2d, 1603-04, 810 F.2d at 1576.

Applicant requests that the rejection of claim 4 be reversed.

Claims 13, 14 and 15 – references fail to disclose or suggest various claimed alternatives for representing a marked node indication

Massalin's binary marker, which (as described above) is used as a lock, is not a pointer. The Office's assertions notwithstanding, there is no disclosure or suggestion that *Massalin's* binary marker, which as described above is used as a lock, ever takes on a value that could be said to be a pointer value, let alone a distinguishing pointer value. Similarly, there is no disclosure or suggestion that *Massalin's* binary marker, is encoded in an otherwise unused portion of a pointer, let alone an unused portion of a next node pointer of a logically deleted node. Finally, there is no disclosure or suggestion in *Massalin* an additional level of indirection used as a marked node indication. Other art of record does not add the missing limitations.

The rejections of claims 13, 14 and 15 are baseless and should be reversed.

Claims 3, 21, and 30 – references fail to disclose or suggest use, as claimed, of CAS or LL/SC synchronization

As reviewed above with regard to claims 1, 16, 25, and 29, the Examiner cannot simply substitute a CAS from *Srinivas* for *Massalin's* CAS2. Since neither *Srinivas* nor *Massalin* disclose use of a CAS operation or LL/SC operation pair as claimed, no *prima facie* case of obviousness has been made out for dependent claims 3, 21 and 30. Other art of record fails to provide the missing limitation(s).

Accordingly, dependent claims 3, 21 and 30 are each allowable for at least this reason as well. Applicant respectfully requests that the rejections be withdrawn and the claims allowed.

CONCLUSION

For the at least the foregoing reasons, Appellants' presently claimed invention would not have been obvious to one of ordinary skill in the art under 35 U.S.C. § 103(a) in view of the cited prior art. Accordingly, this honorable Board is respectfully requested to reverse the rejections of claims 1 – 6 and 8 – 30 and to direct the claims 1-30 of the present application (including allowable claim 7, now rewritten) to be issued.

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that, on the date shown below, this correspondence is being

☒ deposited with the US Postal Service with sufficient postage as first class mail and addressed as shown above.

☐ facsimile transmitted to the US Patent and Trademark Office.

28-Oct-05
Date

EXPRESS MAIL LABEL: _____

Respectfully submitted,

David W. O'Brien, Reg. No. 40,107
Attorney for Applicant(s)
(512) 338-6314
(512) 338-6301 (fax)

CLAIMS APPENDIX

1. A non-blocking concurrent shared object representation comprising:
a linked-list of nodes encoding of a group of zero or more values; and
linearizable operations defined to implement semantics of at least insert and remove operations on the group, wherein concurrent execution of the linearizable operations is mediated using a first synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group, wherein the first synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

2. The non-blocking concurrent shared object representation of claim 1,
wherein concurrent execution of the linearizable operations is further mediated using a second synchronization primitive to physically excise the node corresponding to the logically deleted value.

3. The non-blocking concurrent shared object representation of claim 2,
wherein the first and second synchronization primitives include compare and swap (CAS) operations and load lock/store-and-compare operations.

4. The non-blocking concurrent shared object representation of claim 2,
wherein reclamation of storage associated with the excised node is independent of the linearizable operations.

5. The non-blocking concurrent shared object representation of claim 1,
wherein the linked-list of nodes is free of reference count storage for coordination of garbage collection.

6. The non-blocking concurrent shared object representation of claim 1,
wherein traversal of the concurrent shared object is without atomic update of a garbage collection coordination store.

8. The non-blocking concurrent shared object representation of claim 1,
wherein the node corresponding to the logically deleted value is physically excised from the linked-list by an execution sequence corresponding to one of:
an instance of the remove operation that performed the logical deletion;
an instance of the remove operation that did not perform the logical deletion; and
an instance of the insert operation.

9. The non-blocking concurrent shared object representation of claim 1,
wherein the linearizable operations further implement semantics of a find operation.

10. The non-blocking concurrent shared object representation of claim 1,
wherein the values of the group are stored in respective ones of the nodes.

11. The non-blocking concurrent shared object representation of claim 1,
wherein the values of the group are represented in storage reachable from respective ones of the nodes.

12. The non-blocking concurrent shared object representation of claim 1,
wherein the values of the group are represented in storage identified by respective ones of the nodes.

13. The non-blocking concurrent shared object representation of claim 1,
wherein the marked node indication includes a distinguishing pointer value.

14. The non-blocking concurrent shared object representation of claim 1,
wherein the marked node indication includes a distinguishing bit value in an otherwise unused portion of a next node pointer of the logically deleted node.

15. The non-blocking concurrent shared object representation of claim 1,
wherein respective next node pointers of those of the nodes corresponding to current values of the group directly reference respective other ones of the nodes; and

wherein the marked node indication includes a distinguishing additional level of indirection between the next node pointer of the logically deleted node and a respective other one of the nodes.

16. A method of managing access to a linked-list of nodes susceptible to concurrent operations on a group encoded therein, the method comprising:

separating deletion of a value from the group into at least two functional sequences;

the first functional sequence performing a logical deletion of the value using a synchronization primitive to mark a corresponding one of the nodes; and

the second functional sequence excising the marked node from the linked-list,

wherein the synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

17. The method of claim 16,

wherein the logical deletion and the marked node excision are performed as part of a single deletion operation operating upon the value.

18. The method of claim 16,

wherein the logical deletion is performed as part of a deletion operation operating upon the value; and

wherein the marked node excision is performed as part of another operation.

19. The method of claim 18,

wherein the another operation is an insert operation.

20. The method of claim 18,

wherein the another operation is a remove operation operating upon another node.

21. The method of claim 16,

wherein the synchronization primitive includes a compare and swap (CAS) operation or a load lock/store-and-compare operation.

22. The method of claim 16, further comprising:

after the logical deletion but before the marked node excision, traversing, as part of an access operation, the linked-list including the marked node.

23. The method of claim 16,

wherein the group is an ordered set; and

wherein the linearizable operations implement semantics of a remove operation selective for a value, if any, of the group based on comparison with a specified value.

24. The method of claim 23,

wherein the ordered set is organized in increasing value order; and

wherein the remove operation is selective for a value, if any, of the group greater than or equal to the specified value.

25. A computer program product encoded in at least one computer readable medium, the computer program product comprising:

at least two functional sequences providing non-blocking access to a concurrent shared object, the concurrent shared object instantiable as a linked-list of nodes encoding of a group of zero or more values;

a first of the functional sequences defined to implement semantics of an insert operation on the group; and

a second of the functional sequences defined to implement semantics of a remove operation on the group,

wherein instances of the functional sequences are linearizable and concurrent execution thereof by plural processors of a multiprocessor is mediated using a synchronization primitive to encode a marked node indication signifying logical deletion of a corresponding one of the values from the group with separate physical excision of the corresponding node, wherein the synchronization primitive atomically examines and updates a single target, the updating being conditional on the examination.

26. A computer program product as recited in 25,

embodied as a software component combinable with program code to provide the program code with linearizable, non-blocking access to the concurrent shared object.

27. A computer program product as recited in 25,
embodied as a program executable to provide linearizable, non-blocking access to the concurrent shared object.

28. A computer program product as recited in 25,
wherein the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.

29. An apparatus comprising:
plural processors;
one or more data stores addressable by each of the plural processors;
means for coordinating concurrent execution, by ones of the plural processors, of at least insert and remove operations on a group of zero or more values encoded in the one or more data stores, the coordinating employing a first synchronization primitive to encode an indication signifying logical deletion of a corresponding one of the values from the group and a second synchronization primitive to physically excise the node corresponding to the logically deleted value, wherein the synchronization primitives atomically examine and update their respective single targets, the updating being conditional on the examination; and
means for traversing the encoded group without use of an atomic operation.

30. The apparatus of claim 29 wherein the synchronization primitives include a compare-and-swap operation and a lock load/store-and-compare operations .

EVIDENCE APPENDIX

There is no evidence submitted pursuant to 37 C.F.R. § 1.130, 1.131, or 1.132 or any other evidence entered by the examiner and relied upon by appellant in the appeal.

RELATED APPEALS APPENDIX

There are no decisions rendered by a court or the Board in any proceeding identified above in the Related Appeals and Interferences section.